VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÝ EDITOR AUDIA

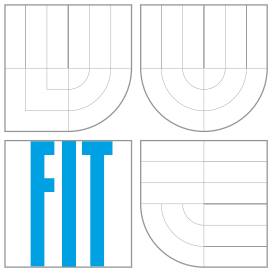
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN MYLER

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÝ EDITOR AUDIA

WEB BASED AUDIO EDITOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN MYLER
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR SCHWARZ, Ph.D.
SUPERVISOR

BRNO 2012

Abstrakt

Tato práce se zabývá tvorbou jednoduchého webového editoru audia s využitím JavaScriptu, HTML5 a nových webových API pro zpracování zvuku (zejména Web Audio API). Popisuje současný stav vývoje a implementace API pro zpracování zvuku v prohlížečích. Dále obsahuje popis návrhu výsledné aplikace až po její implementaci. V závěru práce se nachází shrnutí poznatků získaných při vývoji aplikace a návrh možného budoucího využití a rozšíření.

Abstract

This thesis deals with the creation of simple web-based audio editor using JavaScript, HTML5 and new Web APIs for audio processing (especially the Web Audio API). The thesis describes the current state of development and implementation of APIs for audio processing in browsers. It also contains a description of the resulting application design and its implementation. In the conclusion is a summary of findings of the applications development and proposal of possible future use and expansion.

Klíčová slova

webový editor audia, JavaScript, HTML5, Audio API, Chrome, Backbone.js, web

Keywords

web audio editor, JavaScript, HTML5, Audio API, Chrome, Backbone.js, web

Citace

Jan Myler: Webový editor audia, bakalářská práce, Brno, FIT VUT v Brně, 2012

Webový editor audia

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Schwarze, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Myler
14. května 2012

Poděkování

Děkuji vedoucímu práce Ing. Petru Schwarzovi, Ph.D. za poskytnuté konzultace a odbornou pomoc při vypracování této bakalářské práce. Také děkuji rodině a přátelům za podporu po dobu mého studia.

Obsah

Kapitola 1

Úvod

V současné době lze sledovat narůstající zájem o nové webové technologie, které jsou součástí specifikace HTML5, a ostatní webová API, která jsou intenzivně vyvíjena. Jejich přítomnost umožňuje vývojářům webových aplikací využívat nativních prostředků webových prohlížečů bez nutnosti používání zásuvných modulů a rozšíření třetích stran. Je snaha vytvořit standardizované specifikace, které mohou výrobci webových prohlížečů implementovat, ale také ovlivňovat jejich vývoj spolu s vývojářskou komunitou.

Vzniká stále větší množství webových aplikací, které využívají nativních prostředků prohlížečů a lze je tak využívat napříč širokým spektrem zařízení – od osobních počítačů až po tablety a mobilní telefony.

Cílem zadání této práce byla implementace jednoduchého webového audio editoru, který demonstruje současné možnosti vývoje a nasazení podobných aplikací s využitím čistě nativních vývojářských prostředků. Aplikace byla vytvořena pro prohlížeč Google Chrome, ve které byla také testována.

Kapitola 2 zavádí pojem HTML5 a popisuje historii vzniku až po současnost. V kapitole 3 stručně představím CSS3 a stylovací jazyk LESS. Kapitola 4 shrnuje vývoj JavaScriptu od historie až po dnešní dobu. Dále popisuje JavaScriptové knihovny, které jsem použil při implementaci editoru audia. API pro zpracování zvuku v prohlížečích jsou představeny v kapitole 5. Kapitola 6 obsahuje popis návrhu aplikace, nastínění možností klient/klient-server architektury a prezentaci kladů a záporů obou architektur. Cílem kapitoly 7 je popsat samotnou implementaci a funkce editoru audia. Na závěr v kapitole 8 prezentuji poznatky nabyté z vývoje cílové aplikace, možnosti nasazení do ostrého provozu a možné změny a rozšíření do budoucna.

Kapitola 2

HTML5

V této kapitole stručně představím pojem HTML5, historii vývoje a možnost využití při tvorbě webových stránek a aplikací v dnešní době.

2.1 Historie vzniku HTML5

V roce 1998 se W3C[1] rozhodlo, že vývoj HTML nebude dále pokračovat. Vývoj HTML byl ukončen ve verzi 4.01 a byla představena specifikace XHTML. [5] Což je jednoduše verze HTML4, která – stejně jako XML – vyžaduje striktní syntaktická pravidla (atributy obalené do uvozovek, ukončování všech tagů včetně nepárových, používání malých písmen). Obě specifikace byly nadále hojně využívány.

HTML5 původně vzniklo jako dvě různé specifikace: Web Forms 2.0 a Web Applications 1.0. [3] Obě vyvíjené skupinou WHATWG[2], jež vznikla v roce 2004.

Současně s HTML5 se pracovalo na nové specifikaci XHTML 2.0 (skupina W3C). V roce 2006 W3C uznalo, že bylo příliš optimistické předpokládat, že se začne celosvětově využívat nový formát založený na XML. Nově vytvořená skupina z řad W3C převzala specifikaci od WHATWG jako základ pro novou verzi HTML a započal vývoj dvou oddělených specifikací HTML5. [5]

V roce 2009 byl definitivně ukončen vývoj specifikace XHTML 2.0 a veškerá pozornost byla zaměřena na HTML5. [21]

2.2 Blíže o HTML5

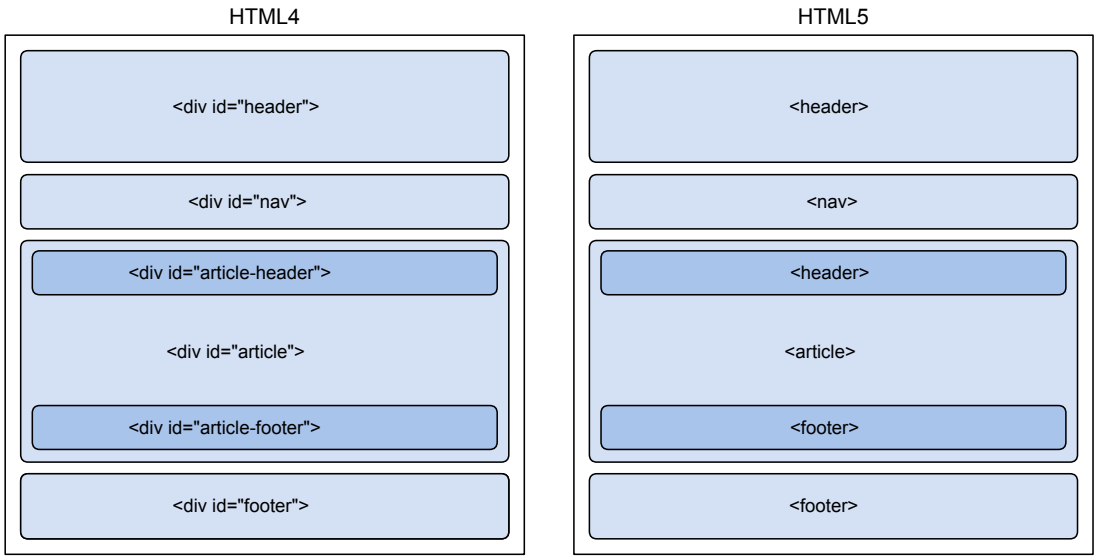
HTML5 není pouze přepracováním předchozích verzí webových značkovacích jazyků. Obsahuje všechny validní elementy z HTML4 i XHTML 1.0. [3] Jazyk již není založený na SGML[3], přesto je zpětně kompatibilní. [15]

Byly znovu definovány existující elementy, u některých byl pozměněn sémantický význam, a také byly – na základě pečlivé analýzy – představeny nové elementy (např. `<nav>`, `<article>`, `<section>`, `<header>`, `<footer>`), které mají zaručit vyšší sémantiku při návrhu webových stránek a aplikací (obr. 2.1).

[1] World Wide Web Consortium, mezinárodní konsorcium, jehož členové za spolupráce veřejnosti vyvíjejí webové standardy.

[2] Web Hypertext Application Technology Working Group, pracovní skupina zajímající se o vývoj a inovace HTML a příbuzných technologií.

[3] Standard Generalized Markup Language, je mezinárodní standard pro popis značkovaného textu. Je to metajazyk pro popis značkovacích jazyků. [4]



Obrázek 2.1: Blokový náhled elementů jednoduché webové stránky

Nespornou výhodou je, že i staré webové stránky můžeme snadno rozšířit a inovovat. Stačí pouze změnit Doctype stránky na `<!DOCTYPE html>` a poté již můžeme přidávat nové prvky z HTML5.

Bylo představeno 13 nových typů formulářových polí (např. `tel`, `range`, `url`, `email`, `datetime`, `color`, `search`, `number`), což opět napomáhá zlepšení sémantiky a použitelnosti dané webové stránky či aplikace. Toho využívají například mobilní zařízení, která pro různé typy formulářových polí automaticky přizpůsobí rozložení klávesnice. Díky nativní implementaci v prohlížečích také postupně odpadá nutnost využívání JavaScriptu pro jejich obsluhu a validaci na straně klienta.

Dále specifikace zahrnuje nové multimediální elementy: `<audio>` a `<video>`. Nové DOM API jako jsou např. Drag and Drop, Web workers, Web storage a Canvas.

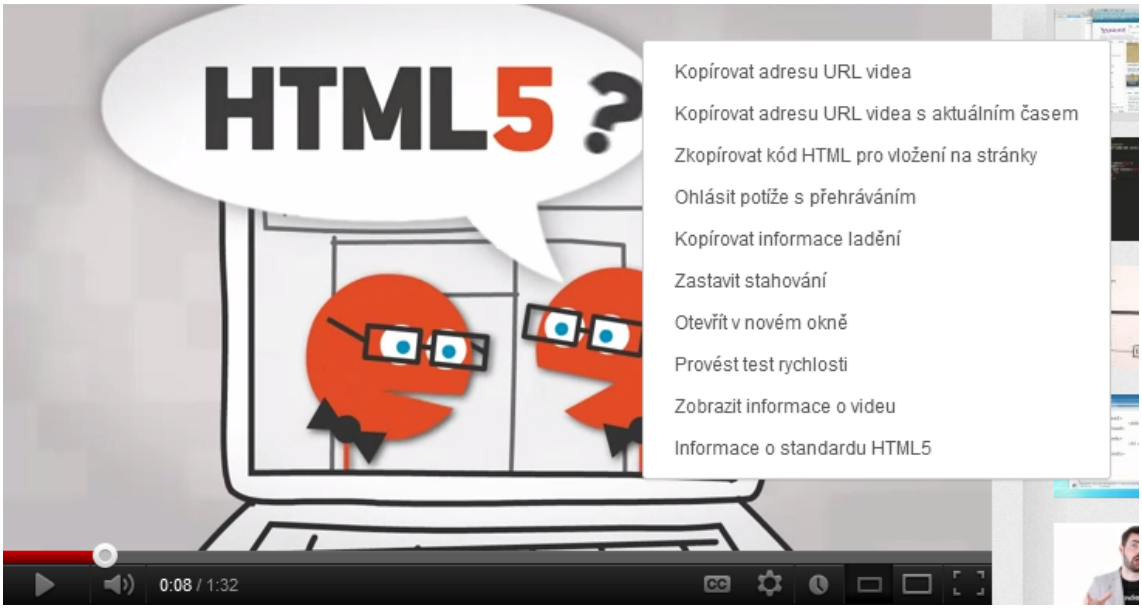
2.3 Současný stav HTML5

Protože na vývoji HTML5 pracují dvě skupiny: W3C a WHATWG, existují dvě odlišné specifikace. Verze spravovaná skupinou W3C je k nalezení na `http://www.w3.org/TR/html5`. Verze skupiny WHATWG se nalézá na `http://whatwg.org/html` (zde došlo k odstranění čísla 5 a výsledný název je pouze HTML).

Obě specifikace jsou velmi podobné, i když verze od WHATWG je více neformální a experimentální. Specifikace jsou zatím pouze ve formě pracovního návrhu[4]. W3C ohlásilo, že v roce 2014 zveřejní verzi specifikace ve fázi W3C doporučení [21], jež má být stabilním a otestovaným doporučením konsorcia.

Přestože ani v současné době není hotová finální verze specifikace, výrobci webových prohlížečů postupně implementují nové vlastnosti HTML5 a webových API. Díky tomu roste komunita vývojářů, kteří nových vlastností využívají pro implementaci webových stránek

[4]Dokumenty W3C mají několik úrovní „zralosti“: pracovní návrh (WD, Working Draft), kandidát na doporučení (CD, Candidate Recommendation), návrh na doporučení (PR, Proposed Recommendation) a nakonec W3C doporučení (REC, W3C Recommendation).



Obrázek 2.2: HTML5 přehrávač youtube.com

a aplikací. Díky webům jako http://caniuse.com nebo http://html5please.com mohou vývojáři sledovat podporu jednotlivých vlastností ve všech rozšířených prohlížečích a informovat se o případné nutnosti použít jiné technologie pro správné fungování v požadovaných prohlížečích.

Narůstající popularitu HTML5 lze sledovat také u známých webových služeb a firem jako jsou např. Google, Facebook, Twitter, Youtube (obr. 2.2) a Vimeo. Všichni z nich již využívají některých vlastností, které HTML5 nabízí. Zejména poté, co společnost Apple přestala ve svých mobilních zařízeních podporovat technologii Adobe Flash, byla pro velké webové služby vytvořena alternativní rozhraní tak, aby využívala právě nových multimediálních prvků HTML5.

Kapitola 3

CSS3 a LESS

CSS (Cascading Style Sheets) je jazyk, který popisuje vizuální formát webových stránek a aplikací. CSS je nezávislé na HTML a může být použito s kterýmkoliv jazykem, založeným na XML. [14]

CSS3 je na rozdíl od verze 2, kterou tvoří jedna samotná specifikace, rozděleno na moduly, které jako základ používají specifikaci CSS2.1. Každý z těchto modulů přidává novou funkcionalitu a/nebo nahrazuje část CSS2.1 specifikace. [12] Rozdělení specifikace také umožňuje vydávání doporučení standardu jednotlivých modulů nezávisle na sobě.

Podobně jako u HTML5 je většina nových vlastností CSS3 zatím specifikována pouze ve fázi pracovního návrhu, ale i přesto jsou nové vlastnosti postupně přidávány do nejnovějších verzí webových prohlížečů.

3.1 Co je LESS?

LESS je dynamický stylovací jazyk. Rozšiřuje CSS o dynamické vlastnosti jako proměnné, tzv. mixiny, vnořená pravidla, operace a funkce. [17]

Jinými slovy je LESS jedním z populárních CSS preprocesorů, které ze stylopisů zapsaných kódem ve vlastní syntaxi vygenerují CSS soubory. Přínosem těchto preprocesorů je snažší udržovatelnost rozsáhlých stylopisů a možnost využití rozšiřujících vlastností, které samotné CSS nenabízí. Za nevýhodu můžeme považovat nutnost překladu zdrojového kódu na CSS stylopis, který probíhá buď na straně klienta pomocí JavaScriptu, nebo na straně serveru – JavaScript nebo např. PHP. Další možností je zdrojové soubory přeložit před zveřejněním produkční verze.

Zdrojový kód 3.1: Ukázka zápisu LESS syntaxe a CSS výstupu

```
// definice promennych
@blue: #00f;
@navWidth: 400px;

// definice mixinu (vcetne vychoziho parametru @shadow)
.box-shadow(@shadow: 0 1px 3px rgba(0,0,0,.25)) {
  -webkit-box-shadow: @shadow;
     -moz-box-shadow: @shadow;
          box-shadow: @shadow;
}

// zanorena pravidla
# header {
```

```
    h1 {
        font-size: 26px;
        color: @blue;
    }
    nav {
        width: @navWidth;
        background: @blue;
        .box-shadow;
    }
}

/*** CSS vystup po prelozeni: ***/
#header h1 {
    font-size: 26px;
    color: #00f;
}

#header nav {
    width: 400px;
    background: #00f;
    -webkit-box-shadow: 0 1px 3px rgba(0,0,0,.25);
       -moz-box-shadow: 0 1px 3px rgba(0,0,0,.25);
            box-shadow: 0 1px 3px rgba(0,0,0,.25);
}
```

Kapitola 4

JavaScript

Cílem této kapitoly je popsat krátce historii JavaScriptu a jeho pozici v dnešní době. Také představím JavaScriptové frameworky, které jsem využil při tvorbě audio editoru a zdůvodním jejich výběr.

4.1 Historie JavaScriptu

JavaScript původně vyvinul Brendan Eich ve společnosti Netscape. [8] Poprvé byl zveřejněn v roce 1995 a nesl označení LiveScript, nedlouho poté byl však přejmenován na JavaScript. Nové jméno bylo zdrojem omylů, jelikož často docházelo ke spojování JavaScriptu s jazykem Java, i když oba jazyky nemají nic společného. Změna jména byla také mnohými označována za chytrý marketingový krok společnosti Netscape: jméno JavaScript využilo podobnosti s – v té době velmi populárním – jazykem Java. [1]

Ve svých počátcích nebyl JavaScript moc populární, byl považován za jazyk nadšenců, kteří jej používali pro vytváření neužitečných grafických efektů, běžících zpráv ve stavové liště prohlížeče apod. JavaScript se také potýkal s problémy způsobenými nekompatibilitami mezi dvěma rozšířenými prohlížeči tehdejší doby: Internet Explorer a Netscape Navigator. Výrobci obou prohlížečů se snažili předhánět tím, že přidávali nové a (zdánlivě) lepší funkce. Prohlížeče se tak často chovaly velmi rozdílně, což ztěžovalo vývoj JavaScriptových programů, které by fungovaly dobře v obou prohlížečích. [6]

4.2 JavaScript dnes

V dnešní době je JavaScript široce rozšířeným a oblíbeným jazykem. I přes standardizaci stále existují malé nekompatibility mezi prohlížeči, avšak při použití některého z JavaScriptových frameworků se nemusíme o tyto odlišnosti starat viz 4.3.

Podobně jako u HTML5, JavaScript značně zpopularizovaly velké společnosti, které pomocí jej začaly vyvíjet bohaté internetové aplikace (např. Gmail). JavaScriptové interprety ve webových prohlížečích jsou již natolik rychlé, že umožňují běh i značně rozsáhlých a výpočetně náročných JavaScriptových aplikací.

JavaScript není využíván pouze pro skriptování na webových stránkách, lze pomocí jej vyvíjet různé widgety, doplňky do webových prohlížečů, mobilní aplikace. Existují také serverová běhová prostředí pro JavaScript (např. Node.js).

4.3 JavaScriptové frameworky

JavaScriptové knihovny a frameworky vznikají z potřeby usnadnit a urychlit vývoj nových aplikací. Oproti samotnému JavaScriptu[1] nabízejí připravené funkce, které mají ošetřeny rozdíly v implementaci mezi prohlížeči. Existuje velké množství různých JavaScriptových frameworků, které se liší svým přístupem, zaměřením, zápisem syntaxe a jazykem v němž jsou implementovány.

jQuery

jQuery je rychlá a malá JavaScriptová knihovna, která zjednodušuje práci s DOM[2] strukturou dokumentu, zpracování událostí, animace a AJAXové[3] interakce. [16]

Tento framework jsem si vybral zejména pro jeho rozšířenost a jednoduché použití při výběru elementů v DOMu, editaci jejich vlastností a také správě událostí. Navíc existuje nepřeberné množství pluginů, které rozšiřují základní funkčnost.

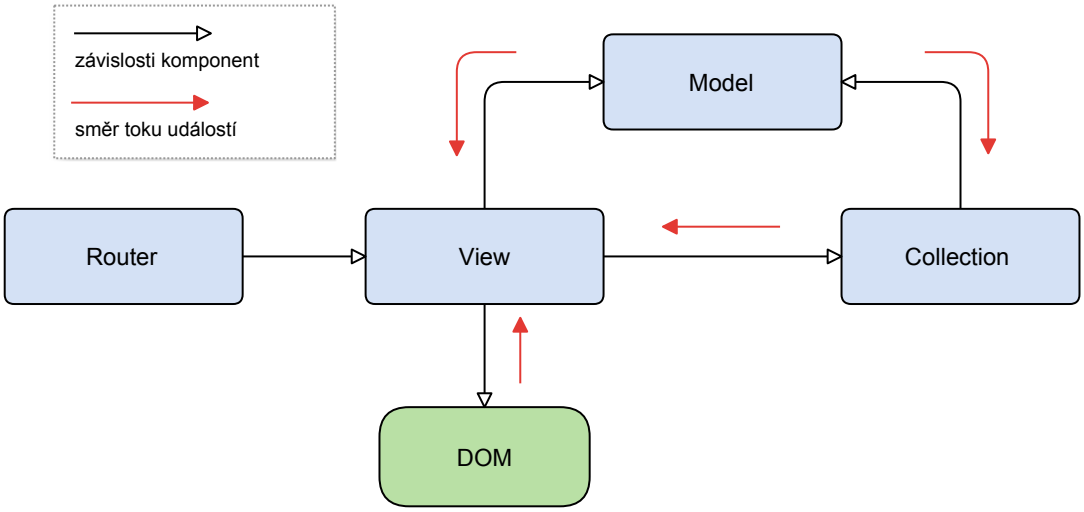
Zdrojový kód 4.1: Ukázka práce s jQuery

```
// spusteni funkce na udalost DOM ready (stejne jako $(document).ready())
$(function() {
    // jQuery selektor vraci jQuery objekt obsahujici elementy,
    // ktere odpovidaji vyrazu uvnitr selektoru

    // vybere vsechny <p> s tridou 'highlight' a ulozi
    // referenci na jQuery objekt
    var $pHighlight = $('p.highlight');

    // vybere <div> s id 'fade-me' a skryje jej s efektem fadeOut
    $('div#fade-me').fadeOut('slow');

    // prida css tridu 'red' vsem vybranym odstavcum,
    // zmeni barvu pozadi na '#000'
    $pHighlight.addClass('red').css('background', '#000');

    // pri kliknuti na vybrany odkaz zobrazi alert okno
    $('a.shout').click(function() {
        alert('Hello world!');
    });

    // vypise text vsech nadpisu druhe urovne do konzole
    $('h2').each(function() {
        // zde 'this' neni jQuery objekt, ale cisty DOM objekt
        console.log(this.textContent);
    });
});
```

[1] V anglické literatuře se často používá pojem *Vanilla JavaScript*.

[2] Document Object Model, je API pro validní HTML a dobře strukturované XML dokumenty. Definuje logickou strukturu dokumentů a způsob, kterým lze přistupovat k jednotlivým objektům dokumentu a pracovat s nimi. [23]

[3] Asynchronous JavaScript and XML, není „oficiální“ technologie jako např. HTML nebo JavaScript. Je to termín, který odkazuje na společnou interakci více technologií – JavaScriptu, webového prohlížeče a webového serveru – vedoucí k získání a zobrazení nových dat bez znovunačtení celé webové stránky. [6]



Obrázek 4.1: Vztahy mezi komponentami Backbone.js

Backbone.js

Následující sekce vychází z [7, 11].

Backbone.js je jeden z mnoha JavaScriptových frameworků pro vytváření webových MV*[4] aplikací.

Umožňuje zejména:

- organizovat strukturu aplikace,
- zjednodušit uchovávání dat na serveru,
- oddělit data aplikace od DOMu,
- synchronizaci mezi DOMem, modely a kolekcemi.

Backbone.js dbá především na potlačení svazování aplikačních dat s DOMem. Data jsou reprezentována v modelech. Při změně dat (vyvolané například akcí z UI[5]) model spustí událost „změna“ (obr. 4.1); všechny pohledy, které zobrazují data z daného modelu mohou na vyvolanou událost zareagovat a například se znovu vykreslit s novými daty. Modely jsou zpravidla uchovávány v kolekcích, což jsou řazené množiny modelů. Také umožňuje zpracování fragmentů URL – routování. Klientská aplikace také může být navázána na existující serverové rozhraní.

Tento framework mi pomohl vytvořit modulární, událostmi provázanou, strukturu aplikace. Která odděluje datovou a prezentační vrstvu, a která umožňuje budoucí rozšíření o nové funkce a prvky.

[4] Jelikož Backbone.js nevykazuje přesně vlastnosti MVC (Model View Controller) ani MVP (Model View Presenter), bývá někdy označován jako MV* architektura. Více viz [7]

[5] User Interface, uživatelské rozhraní aplikace.

Zdrojový kód 4.2: Jednoduchá ukázka Backbone.js

```
// definice modelu
var Person = Backbone.Model.extend({
    defaults: {
        firstName: 'Jan',
        lastName:  'Myler'
    },
    initialize: function() {
        // bindings, validation, inits here
    },
    getName: function() {
        return this.get('firstName')
                + ' ' + this.get('lastName');
    }
});

// definice pohledu
var Name = Backbone.View.extend({
    el: $('#person-name'),
    render: function() {
        this.el.html(this.model.getName());
        return this;
    }
});

// vytvoreni modelu a jeho pohledu
new Name({
    model: new Person({age: 23})
});
```

Require.js

Na rozdíl od většiny ostatních jazyků, JavaScript postrádá vestavěný mechanizmus pro načítání knihoven a modulů. [2] RequireJS je JavaScriptová knihovna, jež umožňuje definování vzájemných závislostí modulů a v případě potřeby zajistí jejich nahrání ve správném pořadí. [20] Díky tomuto lze tvořit aplikace, v nichž se některé moduly zavádí až při jejich potřebě, to může snížit velikost nahrávaných dat při spouštění aplikace.

Další užitečnou vlastností RequireJS je jeho optimalizátor, díky kterému lze sloučit a minifikovat[6] jednotlivé JavaScriptové moduly, ale také CSS soubory. Zmenšení souborů má kladný vliv na velikost dat načítaných prohlížečem, sloučení souborů sníží počet HTTP požadavků na server.

[6] Minifikace je proces odstranění nepotřebných znaků a částí zdrojového kódu beze změny jeho funkcionality.

Kapitola 5

API pro zpracování zvuku

Nové HTML5 elementy `<audio>` a `<video>` umožňují přehrávání existujících audio souborů. Tato funkcionalita však pokaždé není dostačující. Nové API pro zpracování zvuku poskytují funkce pro pokročilé interaktivní aplikace včetně možnosti zpracování a syntézy zvuku řízené přímo ze skriptů aplikace. [10]

V současné době existují dva návrhy specifikace API pro zpracování zvuku: Web Audio API (Google) a MediaStream Processing API (Mozilla). Obě specifikace pokrývají mnoho stejných případů užití, ale v některých oblastech specifikace se vzájemně doplňují. Obrázku 5.1 zobrazuje podporu obou Audio API v prohlížečích.

Obě API nabízí různý přístup k implementaci vlastností jako například: zpracování a vizualizace zvuku v reálném čase, míchání více zvukových zdrojů, změna hlasitosti a vyvážení stran, aplikace filtrů (horní/dolní propust, pásmová propust/zádrž) a efektů.

5.1 MediaStream Processing API

Toto API je ve fázi vývoje a dostupné zdroje jsou pouze ve formě pracovních návrhů. Specifikace slučuje stávající a navrhované funkce zpracovávající média v reálném čase. [18]

MediaStream Processing API je dnes podporováno pouze v prohlížeči Mozilla Firefox. Při vývoji audio editoru jsem jej nepoužil, protože neumožňuje zpracování audia jinak než v reálném čase, což bylo potřebné pro získání dat k vykreslení zvukové stopy.

5.2 Web Audio API

Stejně jako předchozí je Web Audio API ve fázi vývoje a dostupné dokumenty jsou ve formě pracovního návrhu.

Specifikace popisuje vysokoúrovňové JavaScriptové API pro zpracování a syntézu zvuku ve webových aplikacích. Hlavní myšlenkou návrhu je audio graf (obr. 5.2), který obsahuje propojené objekty – uzly audio grafu (`AudioNode`). Propojení uzlů definuje výsledné renderování audia. Samotné zpracování zvuku probíhá v nižší vrstvě, která je implementována optimalizovaným kódem (typicky Assembler, C nebo C++). Přímé zpracování zvuku JavaScriptem je také podporováno. [22]

Audio API - Working Draft

High-level JavaScript API for processing and synthesizing audio

Resources: MediaStream Processing API | Web Audio API by Google | Polyfill to support Web Audio API in Firefox

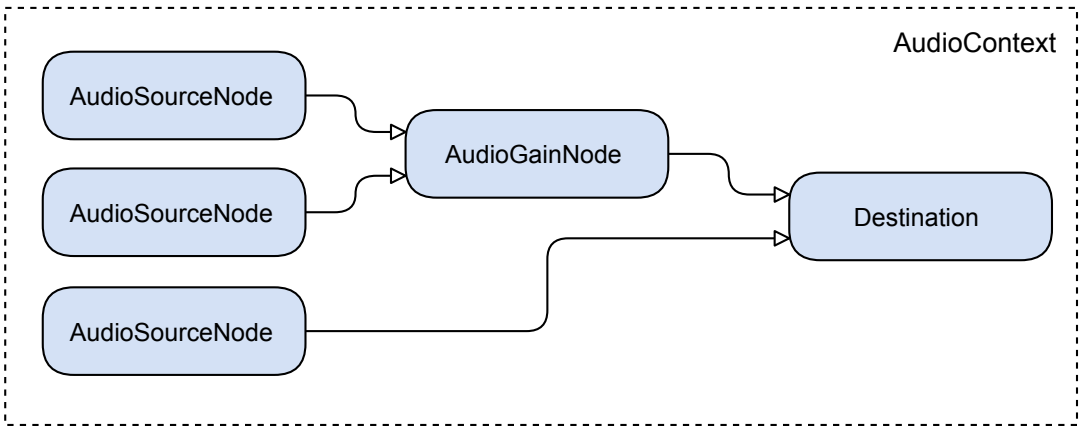
*Usage stats:	Global
Support:	0%
Partial support:	45.23%
Total:	45.23%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
		3.6						10.0	2.1
	6.0	9.0 moz				3.2		11.0	2.2
	7.0	10.0 moz				4.0-4.1		11.1	2.3
	8.0	11.0 moz	17.0	5.0		4.2-4.3		11.5	3.0
Current	9.0	12.0 moz	18.0	5.1	11.6	5.0	5.0-6.0	12.0	4.0
Near future	10.0	13.0 moz	19.0	5.2	12.0				
Farther future		14.0 moz	20.0						

Note: Current support in Gecko/WebKit is based on two different proposals.

Feedback

Obrázek 5.1: Tabulka podpory Audio API v prohlížečích. [24]



Obrázek 5.2: Ukázka audio stromu Web Audio API

Kapitola 6

Návrh aplikace

Při návrhu jsem postupně řešil následující body:

- architekturu aplikace – pouze klient nebo klient-server řešení,
- cílovou platformu – který prohlížeč pro daný operační systém bude aplikaci podporovat,
- definování požadavků na funkce aplikace,
- vytvoření grafického rozhraní,
- zmapování dostupných vlastností existujících Audio API,
- výběr vhodných knihoven a frameworků,
- volbu CVS[1] pro správu projektu.

6.1 Architektura aplikace

Pro architekturu aplikace se nabízí dvě řešení: aplikace kompletně na straně klienta nebo aplikace využívající komunikace se serverovou částí. Obě řešení mají své – mnohdy komplementární – výhody a nevýhody, jejichž výčet bude obecný, bez ohledu na skutečně implementované funkce audio editoru.

Řešení architekturou klient

Toto řešení využívá pro svůj běh pouze webový prohlížeč a je implementováno v JavaScriptu bez využití zásuvných modulů a rozšíření třetích stran (např. Adobe Flash).

Mezi výhody takového řešení patří:

- možnost používat aplikaci i bez připojení k internetu,
- rychlejší zpracování požadavků – potřebná data jsou na straně klienta,
- omezení datových přenosů – důležité zejména pro mobilní platformy.

Mezi nevýhody naopak patří:

[1] Concurrent Version System, je systém, který slouží ke správě verzí (nejen) zdrojových kódů projektu.

- aplikace je závislá pouze na výpočetním výkonu klienta – může být problém na mobilních platformách a starších zařízeních,
- nutnost udržovat načtené zvukové soubory v paměti – vysoká paměťová náročnost aplikace,
- v počátcích návrhu nebylo jasné, zda-li bude možný přímý přístup k datům zvukových souborů (kvůli vykreslování audio stopy),
- není možnost synchronizovat vytvořené projekty mezi více zařízeními.

Řešení architekturou klient-server

Toto řešení počítá s vytvořením serverové aplikace, která bude udržovat audio soubory, zajišťovat jejich analýzu a streamovat data do klientské části.

Výhody této architektury:

- nižší paměťová náročnost klientské části aplikace,
- možnost volby „libovolného“ jazyka (i kompilovaného) pro implementaci serverové části,
- ukládání projektů uživatelů a synchronizace mezi zařízeními.

Nevýhody jsou:

- vysoká náročnost na datové přenosy – problém u mobilních datových tarifů,
- pomalé datové připojení ovlivní rychlost chodu aplikace,
- lokální zvukové soubory se musí nejprve přenést na server,
- aplikaci není možné používat bez připojení k internetu.

Rozhodujícím faktorem pro výběr architektury aplikace byla možnost analyzovat dekódované zvukové soubory rychleji než v reálném čase, což bylo kritické pro vykreslování zvukových stop. Tato funkčnost byla ve Web Audio API zpřístupněna, a proto nebylo nutné použít architekturu klient-server. Pro implementaci pouze na straně klienta jsem se rozhodl také z experimentálního hlediska.

Při návrhu architektury aplikace, z pohledu zdrojového kódu, jsem za hlavní faktor zvolil rozdělení implementace do samostatných modulů dle architektury MV* (viz 4.3). Rozdělení na moduly umožňuje relativně snadno měnit vzhled a rozložení uživatelského rozhraní aplikace bez zásahů do nižší vrstvy funkčního jádra, usnadňuje přidávání nových či změnu stávajích funkcí a správu zdrojových kódů. Moduly jsou rozdělené na tři skupiny: modely a jejich kolekce (udržují stav aplikace), pohledy (prezentují aplikační data uživateli a obsluhují jeho interakci s aplikací), pomocné moduly pro zobrazování a přehrávání.

6.2 Funkce aplikace

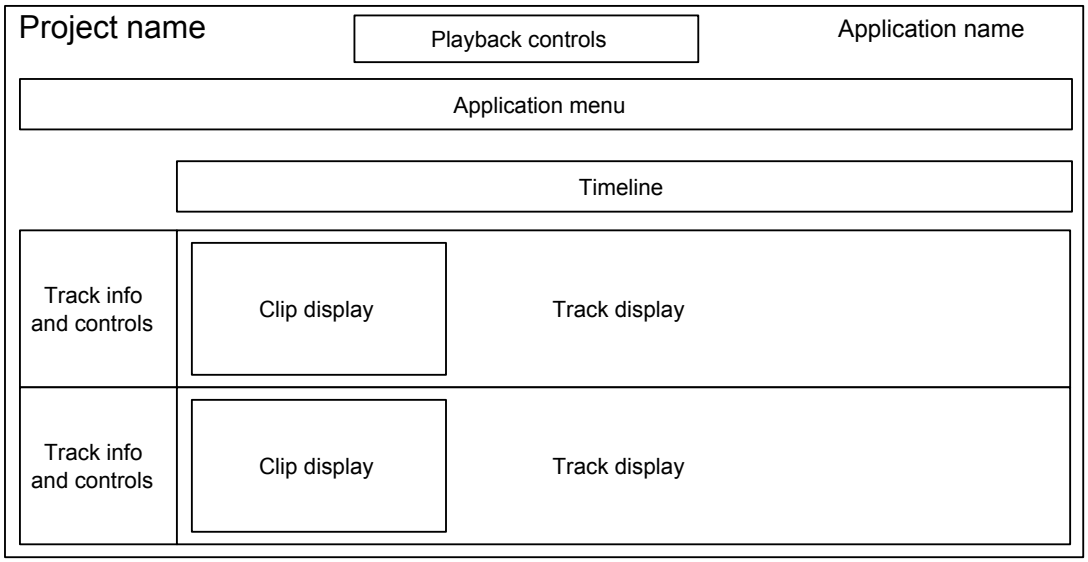
Webový editor audia by měl umožňovat:

- otevření zvukových souborů, které jsou uloženy lokálně na použitém zařízení,

- zobrazení zvukové stopy otevřených souborů,
- přiblížení a oddálení zvukových stop,
- editaci otevřených souborů – kopírování, vyjmutí, vložení a smazání části zvukové stopy,
- nastavení hlasitosti pro jednotlivé zvukové stopy,
- přehrání zvukových stop.

6.3 Návrh grafického rozhraní aplikace

Dalším krokem bylo vytvoření návrhu grafického rozhraní aplikace. Na obrázku 6.1 je zobrazeno jednoduché schéma rozmístění hlavních grafických komponent aplikace. Důležitým požadavkem na grafické rozhraní aplikace je přispůsobivost různým velikostem okna prohlížeče resp. rozlišením obrazovky. V případě nasazení aplikace na zařízení jako jsou tablety by bylo grafické rozhraní patřičně optimalizováno pro pohodlné ovládání pomocí prstů (dostatečně velké ovládací prvky, použití ovládacích gest).



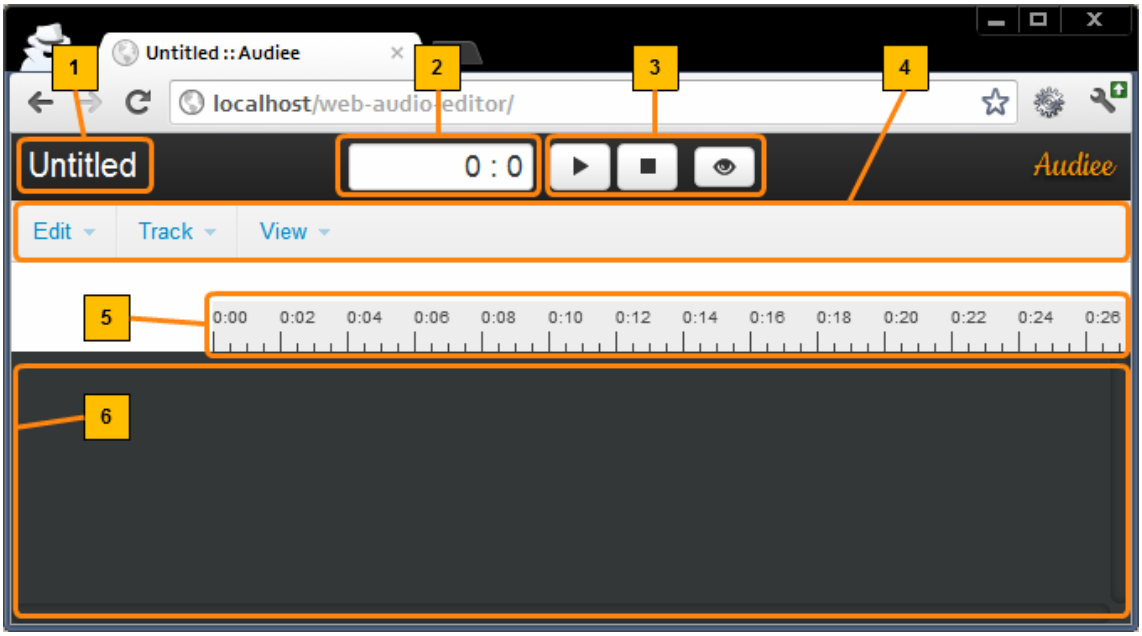
Obrázek 6.1: Zjednodušený návrh rozmístění prvků aplikace

Při návrhu rozhraní jsem se zaměřil především na jednoduchost ovládání. Dostupné akce je možné provádět bez nutnosti výběru příslušných nástrojů jako tomu je v některých desktopových audio editorech. Akce se provádí v rámci daného kontextu – zpravidla v závislosti na pozici programového kurzoru popř. výběru v oblasti zobrazených zvukových stop, jiné akce jsou prováděny přímou interakcí s příslušnými elementy. Většina akcí má také přiřazenu klávesovou zkratku.

Popis grafického rozhraní

Na obrázku 6.2 je zobrazeno okno aplikace po spuštění. Číslo *1* označuje název projektu, v současné verzi aplikace není podporováno ukládání projektů, proto je změna tohoto jména

zablokována. Číslo *2* je zobrazení času přehrávání. Číslo *3* jsou ovládací prvky pro přehrávání. Kliknutí na tlačítko „přehrát“ spustí přehrávání otevřených zvukových stop, buď od začátku, nebo od pozice programového kurzoru. Opětovné kliknutí při probíhajícím přehrávání (tlačítko je oranžové) obnoví přehrávání od předešlé pozice. Kliknutí na tlačítko „zastavit“ zastaví právě probíhající přehrávání. Opětovné stisknutí při zastaveném přehrávání nastaví počáteční pozici na začátek, zároveň dojde ke zrušení pozice kurzoru a deaktivaci vybrané stopy. Spuštění a zastavení přehrávání je také nastaveno na klávesu *mezerník*. Poslední tlačítko se symbolem oka povoluje a zakazuje následování ukazatele pozice přehrávání. Číslo *4* je hlavní menu aplikace. Číslo *5* je časová osa editoru. A nakonec číslo *6* je oblast editoru.



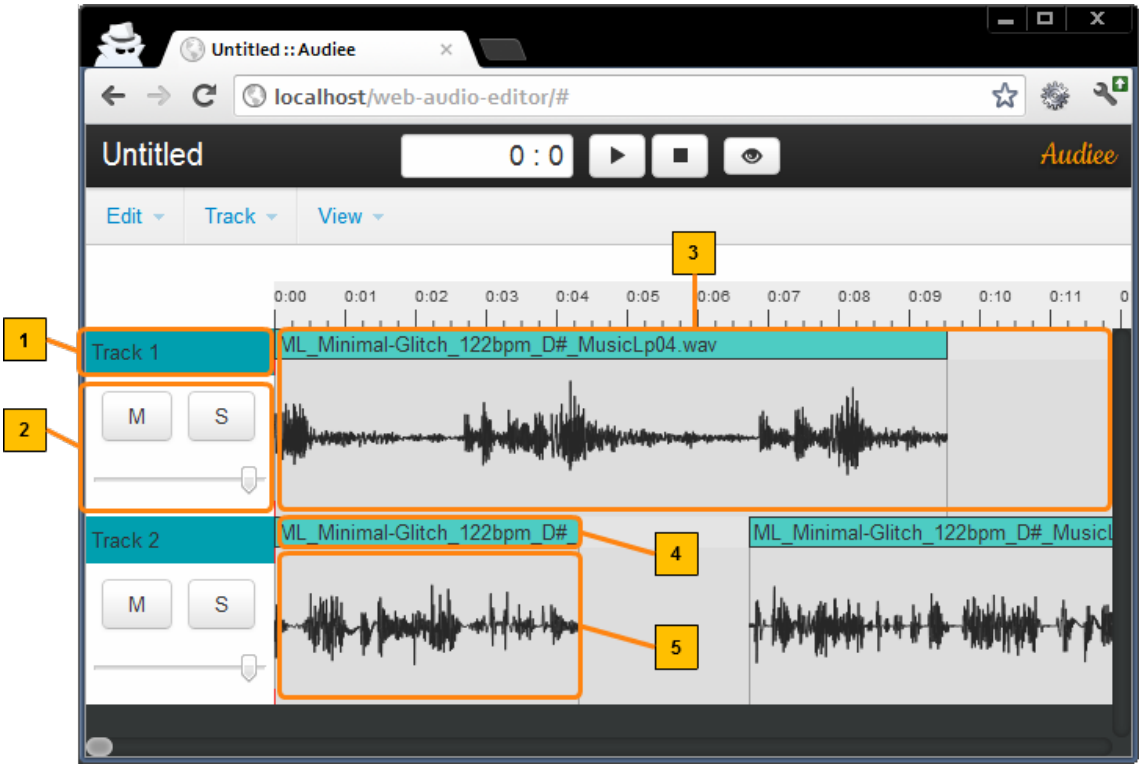
Obrázek 6.2: Vzhled okna editoru po spuštění

Na obrázku 6.3 je zobrazeno okno aplikace po přidání dvou zvukových stop. Číslo *1* značí název zvukové stopy, ten je vygenerován automaticky. Číslo *2* jsou ovládací prvky pro příslušnou stopu, *M* umlčí stopu, *S* zapíná sólo přehrávání a dolní jezdec nastavuje hlasitost stopy. Číslo *3* označuje oblast stopy, ve které se nacházejí jednotlivé zvukové klipy[2]. Číslo *4* označuje název klipu, každý klip má vlastní název, který je vygenerován z názvu zvukového souboru. Názvy stopy i názvy klipů je možné editovat (bude popsáno dále). Číslo *5* značí oblast klipu, ve které je zobrazen odpovídající zvukový úsek.

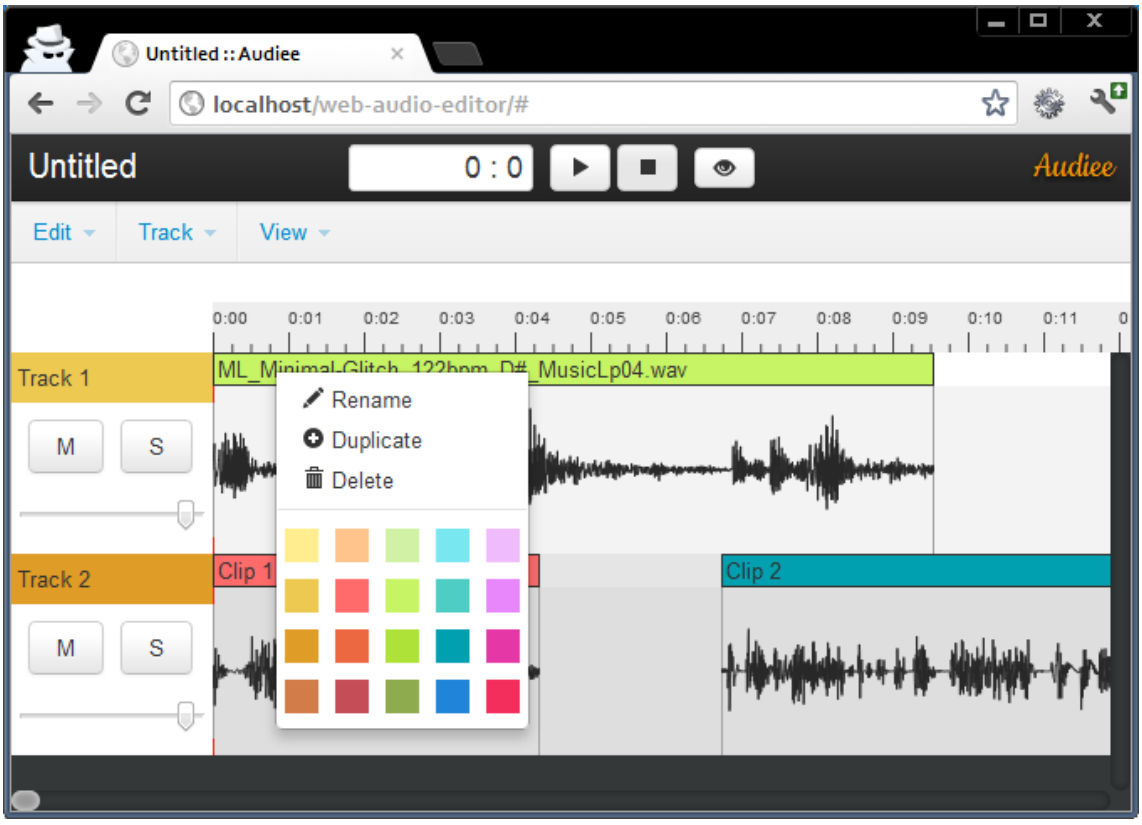
Na obrázku 6.4 je zobrazeno kontextové menu, které se vyvolá kliknutím *pravého tlačítka* myši na oblast názvu stopy nebo klipu. Z tohoto menu je možné vyvolat přejmenování, duplikaci či smazání stopy/klipu. Výběrem barvy z palety ve spodní části menu dojde ke změně pozadí názvu.

Levá část obrázku 6.5 zobrazuje aktivaci stopy – po kliknutí kamkoliv do prostoru stopy se daná stopa aktivuje. To poznáme tak, že se v místě kliknutí vykreslí oranžový kurzor a pozadí stopy se zesvětlí. Aktivní stopu lze smazat použitím hlavního menu (Track – Delete) nebo klávesovou zkratkou `Ctrl + Del`. V pravé části obrázku je znázorněn výběr v rámci

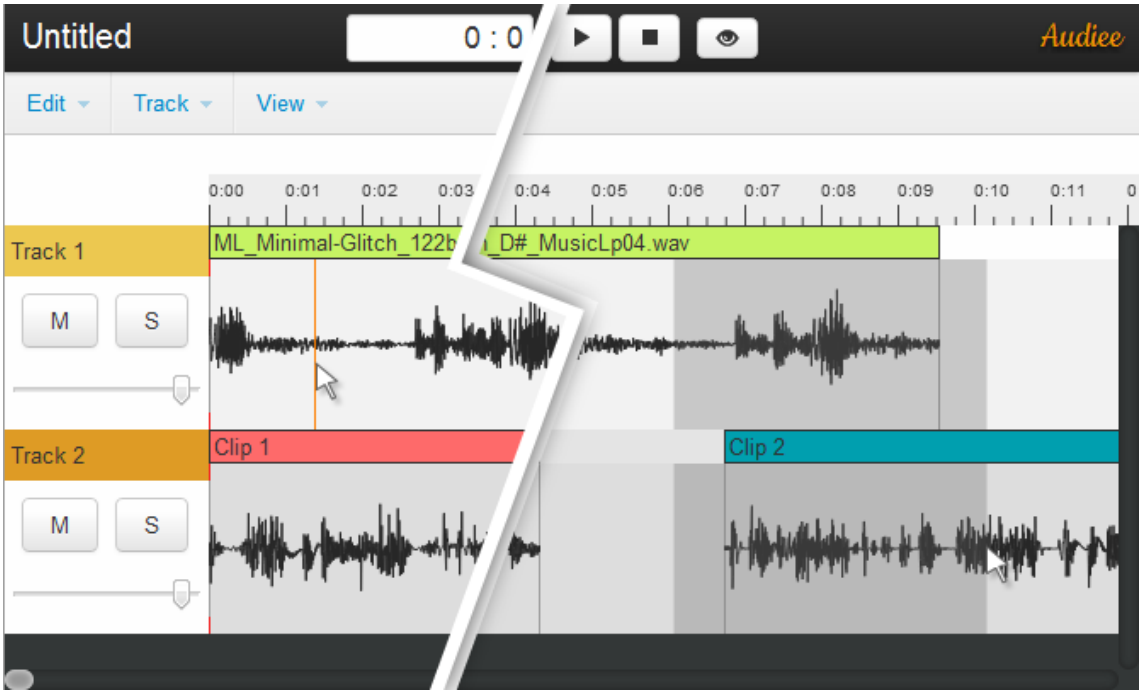
[2]Klip reprezentuje celý nebo část zvukového souboru dané stopy. Editace v rámci stopy se provádí pomocí těchto klipů.



Obrázek 6.3: Vzhled okna editoru po přidání zvukových stop



Obrázek 6.4: Okno editoru se zobrazeným kontextovým menu



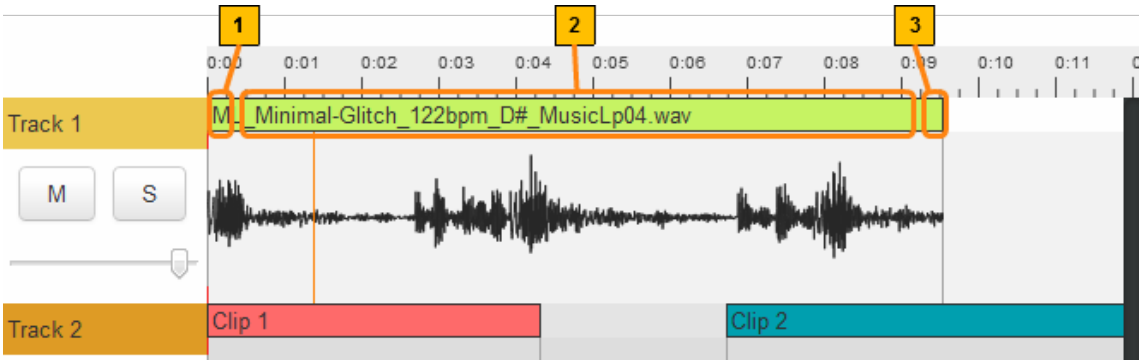
Obrázek 6.5: Ukázka aktivace stopy; kurzoru v rámci stopy (vlevo) a výběru (vpravo)

dvou stop. Výběry lze provádět přes více stop kliknutím do oblasti stopy a tažením myši, uvolněním tlačítka myši nad cílovou stopou se výběr ukončí. Výběr je dále možno upravovat zleva nebo zprava – toho lze docílit stiskem a držením klávesy `shift` při modifikaci.

Zvukové stopy je možné přiblížit a oddálit použitím hlavního menu (View), další možností je držení `Alt` a otáčení kolečkem myši. Držením klávesy `Shift` a otáčením kolečkem myši je možné horizontálně posouvat oblast zvukových stop.

Popis editací zvuku

Jak již bylo zmíněno výše, zvukový soubor každé stopy je reprezentován pomocí jednoho nebo více klipů. Aplikace obsahuje dva způsoby editace klipů. První z nich využívá interakce se samotným klipem (obr. 6.6). Čísla *1* a *3* zvýrazňují oblast, která slouží pro změnu velikosti klipu – kliknutím do dané oblasti a tažením. Číslo *2* ohraničuje oblast sloužící pro změnu pozice klipu – kliknutím do vyznačené oblasti a tažením myší doleva či doprava.



Obrázek 6.6: Úprava vlastností audio klipu

Obrázek 6.7: Ukázka zřetězení audio klipu úpravou jeho velikosti

Druhý způsob editace je založen na práci s kurzorem a výběry, a to tím způsobem, že výběrem označíme požadovanou oblast a na tu se aplikují zvolené akce (z menu Edit) – vyjmutí `X`, zkopírování `C` a smazání `Del`. Po vyjmutí či zkopírování výběru lze obsah schránky vložit na libovolné místo ve stopě, to označíme kurzorem a zvolíme možnost vložit `V`. S pozicí kurzoru pracuje také poslední editace – rozdělení klipu. Kurzorem označíme místo, ve kterém chceme daný klip rozdělit, a použijeme akci rozdělit `E`.

Pokud měníme pozici klipu nebo jeho velikost a výsledek této změny zasahuje do pozice jiných klipů, jsou tyto klipy oříznuty popř. smazány tak, aby nedocházelo k nežádoucímu překryvu klipů.

Na obrázku 6.7 je zobrazeno zřetězení audio klipu. Pokud při editaci velikosti dojde k překročení počátku či konce klipu, automaticky se přidá další – místo návaznosti je označeno dvěma červenými čárkami.

Editace klipů a stop provedené při spuštěném přehrávání (obr. 6.8) se neprojeví do právě přehrávaného zvuku, pro propagaci změn je nutné znovu spustit přehrávání.

6.4 Cílová platforma

Za cílovou platformu jsem zvolil prohlížeč Google Chrome – zpočátku bylo nutné používat vývojářskou verzi, která obsahovala nové funkce Audio API – a operační systém Microsoft Windows. Protože je však Web Audio API implementováno do vykreslovacího jádra WebKit, do budoucna lze očekávat dostupnost funkcí API také v prohlížeči Safari a podobně na Android zařízeních, což by umožňovalo využívat aplikaci na široké škále zařízení.

Obrázek 6.8: Ukázka vzhledu aplikace při přehrávání

6.5 Systém pro správu verzí

Po zvážení dostupných možností a dosavadních zkušeností se systémy SVN[3] a Mercurial[4] jsem se rozhodl vyzkoušet v dnešní době populární verzovací systém GIT[5]. Využil jsem tedy projektu GitHub, který mj. nabízí i propracované webové rozhraní, kde mi po schválení žádosti umožnili vytvořit privátní repozitář pro vzdělávací účely.

[3]Subversion je open source systém pro správu verzí. [9]

[4]Mercurial je zdarma dostupný, distribuovaný systém pro správu verzí. [19]

[5]GIT je zdarma dostupný, distribuovaný systém pro správu verzí navržený pro využití od malých po velmi velké projekty [13]

Kapitola 7

Implementace aplikace

V této kapitole je popsána implementace jednotlivých modulů aplikace a vysvětlení jejich funkcí v rámci aplikace. Jelikož jsou zdrojové kódy modulů a funkcí v nich obsažených mnohdy rozsáhlé, nebude tato kapitola až na výjimky obsahovat ukázky těchto kódů. Zdrojové kódy všech modulů jsou dostupné na přiloženém CD nosiči.

Aplikace po spuštění propaguje pouze jediný objekt `Audiee` do globálního jmenného prostoru, to značně omezuje možnost kolize jmen objektů a proměnných aplikace s jinými objekty a funkcemi. Na obrázku 7.1 je znázorněno schéma hlavního objektu aplikace a vazby mezi jednotlivými moduly.

Všechny moduly aplikace, až na dva pomocné, jsou implementovány pomocí JavaScriptové knihovny Backbone.js (viz 4.3). Struktura objektů, které reprezentují jednotlivé moduly, vychází ze specifikace Backbone.js. Tyto moduly obsahují definice objektů daného typu (model, kolekce, pohled) a před samotným využitím v aplikaci je nutné vytvořit jejich instanci.

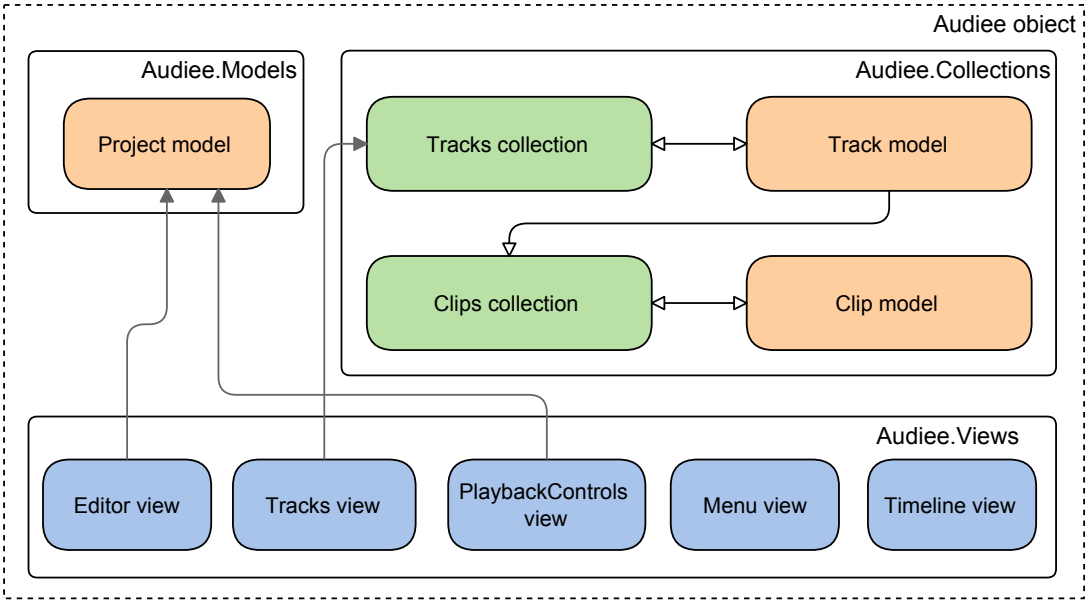
7.1 Modely a kolekce

V této sekci jsou popsány všechny moduly, které reprezentují modely a jejich kolekce. Tyto moduly udržují data aplikace, která by bylo nutné uchovat v případě podpory ukládání vytvořených projektů ať už přímo na lokálním zařízení nebo na serveru.

Atribut `defaults` v modelech slouží pro vytvoření patřičných atributů a nastavení jejich hodnoty při vytvoření instance objektu daného modulu. Funkce `initialize()` slouží např. pro kontrolu validity hodnot vkládaných atributů při vytváření instance objektu, nastavení posluchačů (listenerů) událostí, popř. volání dalších funkcí potřebných pro inicializaci.

Projekt

Model projektu `/js/Audiee/Models.Project.js` uchovává data související se samotným projektem.



Obrázek 7.1: Schéma hlavního objektu aplikace

Zdrojový kód 7.1: Ukázka kódu modulu Models.Project.js

```
var Project = Backbone.Model.extend({
    defaults: {
        name: 'Untitled',
        created: Date.now(),
    },
    initialize: function() { // ... },
    validate: function(attribs) { // ... }
});
```

Tento modul je spíše přípravou pro možné budoucí rozšíření aplikace, jelikož je zatím využíván pouze pro uchování jména nově vytvořeného projektu.

Stopa

Model stopy `/js/Audiee/Models.Track.js` uchovává data jednotlivých zvukových stop. V ukázce zdrojového kódu 7.2 je zároveň ukázána definice závislostí na dalších modulech pomocí Require.js (viz 4.3).

Zdrojový kód 7.2: Ukázka kódu modulu Models.Track.js

```
define([    // definice zavislosti
    'underscore',
    'backbone',
    'Audiee/Collections.Clips',
    'Audiee/Models.Clip'
], function(_, Backbone, ClipsC, ClipM) { // predani zavislosti
var Track = Backbone.Model.extend({
    defaults: {
        name: 'Untitled',
        color: '#00a0b0',
```

```
            gain: 1,
            muted: false,
            solo: false,
            length: 1920 // 32 minutes (default length)
        },
        initialize: function() {
            _.bindAll(this, 'initClip', 'remove', 'resetDuplicates');
            this.bind('remove', this.remove);
            this.bind('change:name', this.resetDuplicates);
            this.clips = new ClipsC;
            this.initClip();
            Audiee.Player.initTrack(this.cid);
        },
        initClip: function() {
            var clip = new ClipM({
                name: this.get('file').name,
                endTime: this.get('buffer').duration,
                buffer: this.get('buffer')
            });
            this.clips.add(clip);
        },
        remove: function() { // ... },
        getSnapshot: function(from, to) { // ... },
        deleteSelection: function(from, to, except) { // ... },
        pasteSelection: function(position, clipboard) { // ... },
        resetDuplicates: function() { // ... }
    });

    return Track;
    });
```

Require.js se postará o nahrání potřebných závislostí, jejichž reference jsou předány do modulu pomocí parametrů funkce, ve které je lze využívat. Ve funkci `initialize` je vidět nastavení posluchačů událostí `this.bind('event', function)`, které objekty Backbone samy vyvolávají při provedení příslušné akce – zde odstranění modelu stopy a změna jména. Ostatní funkce pracují nad daty modelu: `getSnapshot()` získá data výběru (např. pro kopírování/vyjmutí), `deleteSelection()` a `pasteSelection()` vymaže nebo vloží klipy ve vybrané oblasti.

Klip

Model klipu `/js/Audiee/Models.Clip.js` uchovává data audio klipu.

Zdrojový kód 7.3: Ukázka kódu modulu Models.Clip.js

```
var Clip = Backbone.Model.extend({
    defaults: {
        name: 'untitled',
        color: '#4ecdc4',
        trackPos: 0,
        startTime: 0,
        endTime: 0,
        loop: 0
    },
    initialize: function() { // ... },
    clipLength: function() {
        return this.get('endTime')
```

```
            - this.get('startTime')
            + this.get('loop')
            * this.get('buffer').duration;
    },
    duplicate: function() {
        var newClip = this.clone(),
            newTrackPos = newClip.get('trackPos') + this.clipLength();
        newClip.set('trackPos', newTrackPos);
        this.collection.addDuplicate(newClip);
    }
});
```

Každý klip si uchovává jeho pozici `trackPos` v rámci zvukové stopy, začátek `startTime` a konec `endTime` přehrávání zdrojového zvukového souboru a počet opakování klipu `loop` (při zřetězení klipu). Dále je ukázán kód funkce pro získání délky klipu `clipLength()` a duplikaci klipu `duplicate()`.

Stopy

Kolekce stopy `/js/Audiee/Collections.Tracks.js` uchovává množinu modelů zvukových stop. Při vytvoření nové stopy se nový model vloží do této kolekce.

Klipy

Kolekce klipů `/js/Audiee/Collections.Clips.js` uchovává množinu modelů klipů. Každý model zvukové stopy obsahuje jako jeden z atributů právě tuto kolekci.

7.2 Pohledy

Pohledy v naprosté většině reprezentují data modelů nebo kolekcí. Samy reagují na sledované události příslušejících objektů a zpracovávají akce vyvolané uživatelem aplikace. Každý pohled má přidělený právě jeden element ve struktuře stránky, do kterého vykresluje svůj obsah.

Všechny pohledy obsahují dvě základní funkce: `initialize()` zpravidla obsahuje nastavení posluchačů událostí a případné inicializace atributů pohledu, `render()` se stará o vykreslení obsahu pohledu – nejčastěji pomocí připravené šablony pro konkrétní pohled.

Editor

Pohled editoru `/js/Audiee/Views.Editor.js` zajišťuje vykreslení oblasti, do které jsou vkládány zvukové stopy. Reaguje na události změny okna a nastavuje velikost vykreslované oblasti tak, aby vyplňovala celou dostupnou šířku a výšku okna. Dále obsluhuje události skrolování (propaguje událost do dalších pohledů) a použití kolečka myši (vyvolá přiblížení nebo oddálení stop).

V atributech objektu také uchovává informace o aktivní stopě, výběru, pozici kurzoru a schránce výběru. Tyto informace není třeba uchovávat v modelu, protože se nejedná o perzistentní data.

Menu

Pohled menu `/js/Audiee/Views.Menu.js` vykresluje hlavní menu aplikace a zpracovává události s ním spojené. Při kliknutí na položky menu vyvolá příslušnou funkci, také reaguje na stisknuté klávesy (obsluha klávesových zkratek).

Časová osa

Pohled časové osy `/js/Audiee/Views.Timeline.js` má jedinou funkci – vykreslit časovou osu editoru. Pohled reaguje na změny velikosti okna, horizontální skrolování panelu editoru a změnu přiblížení/oddálení stop. Při kterékoliv z těchto událostí se pohled překreslí.

Ovládací prvky přehrávání

Pohled `/js/Audiee/Views.PlaybackControls.js` vykresluje ovládací prvky a zobrazuje čas přehrávání. V případě obsluhy interakcí s ovládacími prvky volá příslušné funkce pomocného modulu pro přehrávání.

Editovatelné jméno

Pohled `/js/Audiee/Views.EditableName.js` je využit pro zobrazení jména projektu, zvukových stop a klipů. Zapouzdřuje zobrazení a obsluhu kontextového menu, které je možné vyvolat u stop a klipů.

Při úpravě jména se pohled změní na editovatelné pole; po skončení úpravy se aktualizuje zobrazované jméno. Změny jména popř. barvy pozadí se ukládají do souvisejících modelů.

Klipy

Pohled klipy `/js/Audiee/Views.Clips.js` vykresluje obsah kolekce klipů. Při přidání nebo odstranění klipu v kolekci dochází k překreslení obsahu příslušného elementu.

Klip

Pohled klip `/js/Audiee/Views.Clip.js` reprezentuje data modelu klipu. Reaguje na změny pozice či počátečního nebo koncového času v modelu. Stejně tak naslouchá událostem přiblížení/oddálení zvukových stop – dochází k překreslení pohledu.

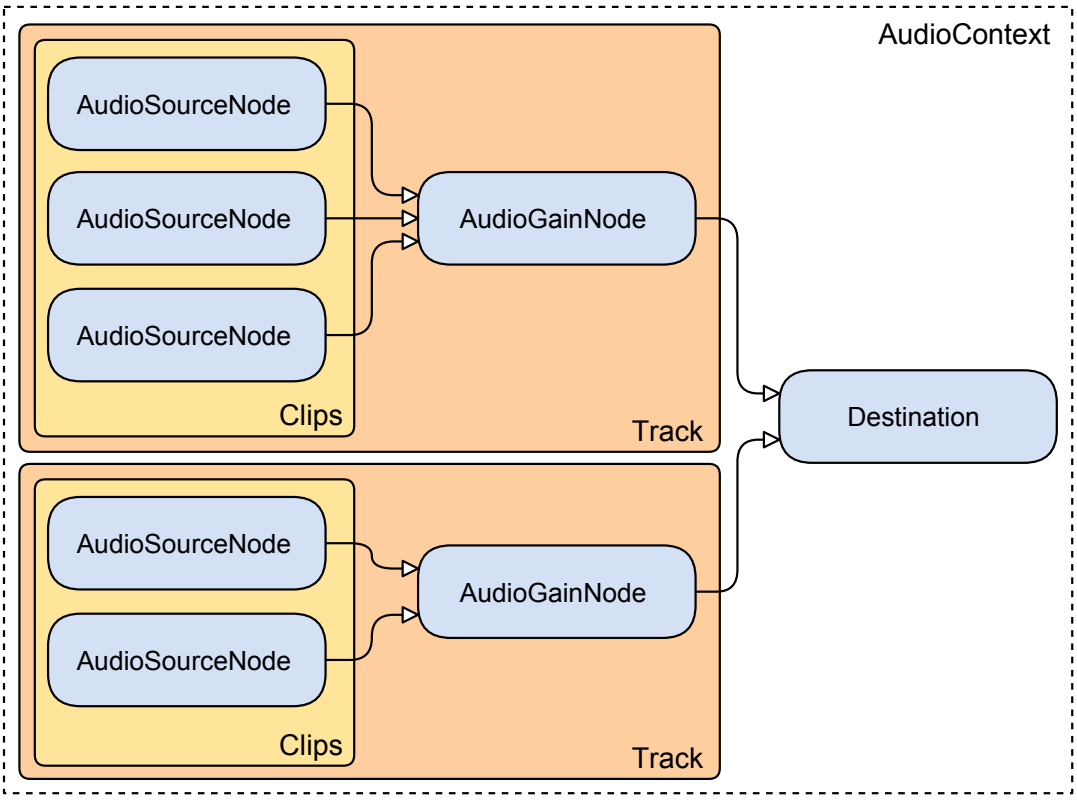
Obsahuje dva vnořené pohledy – editovatelné jméno a pohled pro vykreslení zvukové stopy klipu `/js/Audiee/Views.ClipDisplay.js`, který zobrazuje množinu elementů `<canvas >`, na které se vykresluje zvuková stopa. Samotné vykreslení má na starost pomocný modul pro vykreslování.

Stopy

Pohled stopy `/js/Audiee/Views.Tracks.js` vykresluje obsah kolekce stop. Při přidání nebo odstranění stopy v kolekci dochází k překreslení pohledu.

Stopa

Pohled stopa `/js/Audiee/Views.Track.js` reprezentuje data modelu stopy. Reaguje na přiblížení/oddálení stop změnou velikosti přiřazeného elementu.



Obrázek 7.2: Schéma sestaveného audio stromu pro dvě zvukové stopy

Obsahuje tři vnořené pohledy – editovatelné jméno, ovládací prvky stopy `/js/Audiee/Views.TrackControls.js` a pohled pro vykreslování kurzoru a výběru nad danou stopou `/js/Audiee/Views.TrackDisplay.js`.

7.3 Pomocné moduly

Součástí implementace jsou dva pomocné moduly. Jejich funkce přímo nesouvisí se strukturou aplikace dle MV* architektury.

Přehrávání

Pomocný modul pro přehrávání `/js/Audiee/Helpers.Player.js` obsahuje funkce související s přehráváním. Jako jediný modul přímo pracuje s audio kontextem Web Audio API. Jednou z hlavních funkcí je sestavení audio stromu (obr. 7.2) a zahájení přehrávání zvukových stop.

Zobrazování

Pomocný modul pro zobrazování `/js/Audiee/Helpers.Display.js` obsahuje funkce, které souvisí se zobrazováním a vykreslováním v rámci aplikace. Udržuje úroveň přiblížení stop a obsahuje pomocné funkce, které konvertují sekundy na pixely a opačně.

Mezi vykreslovací funkce patří vykreslení zvukové stopy, vykreslení kurzoru a výběru a zobrazení pozice při přehrávání.

7.4 Problémy při implementaci

Při implementaci jsem narazil na dva hlavní problémy. První z nich se týká HTML5 elementu `<canvas>`, který má definovánu maximální možnou velikost. Pokud je rozměr elementu větší, vykreslená data se neprojeví. Protože při větších úrovních přiblížení překročila šířka canvasu tuto maximální hodnotu, bylo nutné vykreslovací oblast rozdělit na několik menších a programově kontrolovat správné vykreslování dat mezi nimi.

Druhý problém stávající implementace souvisí s pamětí. Otevřené zvukové soubory jsou totiž nahrány do paměti, dekódovány pomocí Web Audio API a dále používány pro vykreslování a přehrávání. Dekódovaná data musí být v paměti po celou dobu běhu aplikace, což značně zvyšuje paměťovou náročnost aplikace. Může se dokonce stát, že potřebná paměť překročí velikost paměti přidělené pro danou záložku prohlížeče a dojde k ukončení chodu celé záložky.

Kapitola 8

Závěr

Cílem této práce bylo vytvořit jednoduchý webový editor audia, při jehož implementaci bylo využito JavaScriptu a nově vytvářených Audio API. Výsledně vytvořená aplikace slouží především jako demonstrace možností, které lze v současné době pomocí Web Audio API implementovat. Stávající aplikace lze využít jako základ pro nové implementace s různým zaměřením využití.

Zejména kvůli stále nízké podpoře nově vznikajících Audio API v prohlížečích (obr. 5.1) nelze vytvořený editor nasadit do ostrého provozu. Problémy s kompatibilitou by snad šlo vyřešit přidáním modulu, který by využíval např. Adobe Flash pro přehrávání a získání audio dat v prohlížečích, které nepodporují nové Audio API.

Další změnou ve stávajícím editoru by mělo být přepracování vykreslování zvukových stop tak, aby byly vykreslovány pouze viditelné části. Stávající implementace vykresluje stopy v celé délce, což při vyšších úrovních přiblížení zpomaluje odezvu aplikace. Další zajímavou možností je využít Web Workers[1], které však z důvodů bezpečnosti nemají přístup k DOM struktuře stránky. S hlavním skriptem komunikují pomocí zasílaných zpráv, takže pro přímé vykreslování je (zatím) nelze použít.

Posledním problémem, který by bylo nutné vyřešit před ostrým provozem aplikace, je vysoká paměťová náročnost kvůli uchovávání dekódovaných audio dat v paměti prohlížeče. Dekódovaná data jsou kromě přehrávání používána pro vykreslování zvukových stop, proto je k nim nutný rychlý přístup po celou dobu běhu aplikace. Paměťová náročnost je však velmi limitující především pro zařízení jako jsou tablety, u kterých je operační paměť nižší.

Rozšíření do budoucna

Jednou z nových funkcí by mohl být například souborový prohlížeč, který by umožňoval snadnější přidávání zvukových souborů do aplikace. Ten by bylo možné vytvořit za pomoci nových File API, která umožňují přístup na disk.

Další funkcí by mohl být panel mixéru, kde by bylo možné ovládat zvukové parametry všech stop. Také možnost aplikovat různé efekty či filtry na zvukové stopy by jistě byla využitelná.

Poslední důležitou funkcí, je možnost ukládání vytvořených projektů, export zpět do zvukového souboru a podpora historie editací, která značně usnadní práci s aplikací.

[1] Web Workers jsou API, které umožňuje spustit JavaScriptový kód umístěný v odděleném souboru. Dojde tak k vytvoření nového vlákna, které je nezávislé na vykonávání hlavního kódu aplikace.

Literatura

[1] Flanagan, David. *JavaScript: The Definitive Guide.* Fifth edition. Sebastopol, CA: O'Reilly, 2006, ISBN 978-0-596-10199-2.

[2] Gauthier, Nick a Chris Strom. *Recipes with Backbone* [online]. 2012. Dostupné z: http://recipeswithbackbone.com/

[3] Goldstein, Alexis, Louis Lazaris a Estelle Weyl. *HTML5 & CSS3 For The Real World.* Collingwood, Australia: SitePoint, 2011, ISBN 978-098-0846-904, 1–10 s.

[4] Hruška, Tomáš a Radek Burget. *Internetové aplikace (WAP) II.: část SGML, HTML, CSS, DOM* [online]. FIT VUT v Brně, 2007, 16–17 s. Dostupné z: https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaWAP2SGMLHTMLCSSDOM.pdf

[5] Lawson, Bruce a Remy Sharp. *Introducing HTML5.* Berkeley, CA: New Riders, 2011, ISBN 978-0-321-68729-6.

[6] McFarland, David Sawyer. *Javascript: The Missing Manual.* Sebastopol, CA: Pogue Press/O'Reilly, 2008, ISBN 978-0-596-51589-8.

[7] Osmani, Addy. *Developing Backbone.js Applications* [online]. 2011. Dostupné z: http://addyosmani.github.com/backbone-fundamentals/

[8] Severance, Charles: JavaScript: Designing a Language in 10 Days. *Computer* [online], 2012, roč. 45, č. 2, s. 7-8 [cit. 2012-05-05], ISSN 0018-9162. DOI: 10.1109/MC.2012.57. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6155645

[9] Apache Subversion [online]. © 2011 [cit. 2012-05-08]. Dostupné z: http://subversion.apache.org/

[10] Audio Processing API. In: *World Wide Web Consortium (W3C)* [online]. 15.12.2011 [cit. 2012-05-06]. Dostupné z: http://www.w3.org/TR/audioproc/

[11] Backbone.js [online]. © 2010-2012 [cit. 2012-05-05]. Dostupné z: http://backbonejs.org/

[12] Cascading Style Sheets (CSS) Snapshot 2010: CSS Level 3. In: *World Wide Web Consortium (W3C)* [online]. 12.05.2011 [cit. 2012-05-04]. Dostupné z: http://www.w3.org/TR/css-2010/#css3

[13] Git [online]. [cit. 2012-05-08].
Dostupné z: http://git-scm.com/

[14] HTML & CSS: What is CSS?. In: *World Wide Web Consortium (W3C)* [online]. © 2012 [cit. 2012-05-04].
Dostupné z: http://www.w3.org/standards/webdesign/htmlcss#whatcss

[15] HTML 5 Reference: A Web Developer's Guide to HTML 5. In: *World Wide Web Consortium (W3C)* [online]. 2009-03-23 [cit. 2012-05-04].
Dostupné z: http://dev.w3.org/html5/html-author/

[16] jQuery: The Write Less, Do More, JavaScript Library [online]. © 2012 [cit. 2012-05-05].
Dostupné z: http://jquery.com/

[17] LESS: The Dynamic Stylesheet language [online]. © 2010-2012 [cit. 2012-05-04].
Dostupné z: http://www.lesscss.org/

[18] MediaStream Processing API. In: *World Wide Web Consortium (W3C)* [online]. 15.12.2011 [cit. 2012-05-06].
Dostupné z: http://www.w3.org/TR/streamproc/

[19] Mercurial SCM [online]. [cit. 2012-05-08].
Dostupné z: http://mercurial.selenic.com/

[20] RequireJS: A JavaScript Module Loader [online]. © 2011 [cit. 2012-05-05].
Dostupné z: http://requirejs.org/

[21] W3C Confirms May 2011 for HTML5 Last Call, Targets 2014 for HTML5 Standard. In: *World Wide Web Consortium (W3C)* [online]. [Feb 2011][cit. 2012-05-04].
Dostupné z: http://www.w3.org/2011/02/htmlwg-pr.html

[22] Web Audio API. In: *World Wide Web Consortium (W3C)* [online]. 15.03.2012 [cit. 2012-05-07].
Dostupné z: http://www.w3.org/TR/webaudio/

[23] What is the Document Object Model? In: *World Wide Web Consortium (W3C)* [online]. 2000-11-13 [cit. 2012-05-05].
Dostupné z:
http://www.w3.org/TR/DOM-Level-2-Core/introduction.html

[24] When can I use... [online]. Apr 2012 [cit. 2012-05-07].
Dostupné z: http://caniuse.com/#feat=audio-api

Příloha A

Obsah CD

Na přiloženém CD nosiči se nachází:

- text bakalářské práce ve formátu PDF,
- text bakalářské práce ve formátu LaTeX,
- zdrojové kódy aplikace ve vývojové verzi,
- zdrojové kódy aplikace v produkční verzi.